WIP: The Optimization of Alcoholism under a Hypothetical Bartering System

Kevin Palani and Kevin Zheng

August 4, 2019

Contents

1 Introduction to the hypothetical bartering system

You have \$10, and a beer is 2\$. Very quickly you can see that if you spend all 10\$, you will get 5 beers. Once you've drunken the 5 beers, you are left with 5 beer bottles and 5 caps. The store owner strikes you a deal. If you give him two empty bottles or four bottle caps, he'll give you a new bottle of beer. He is also kind enough to let you drink before you pay, but you are not allowed to be in bottle or cap-debt. How many drinks you can get?

2 Naive solution

A naive solution would be to consider the worth of the beer itself. Since 2 bottles can get a \$2 beer, that means a bottle is worth \$1. Since 4 caps can get a \$2 beer, that means a cap is worth \$0.50. A beer is consisted of a bottle, a cap, and some drink. Therefore the drink itself is worth \$0.50. If we start with \$10, we can get 20 drinks.

The problem with this solution is that this assumes that in the end we can successfully convert all of our bottles and caps into booze, but can we? Short answer is, no. The only way to successfully convert all of our bottles and caps into booze is if adding exactly one bottle and one cap to our balance would cover the next drink i.e. drink before you pay and the resulting bottle and cap covers your tab perfectly. Which is impossible because there doesn't exist a transaction which takes a bottle and cap at the same time.

Rather, it's still fun to see if we can predict exactly how much we are left with.

3 Modeling caps and bottles

Suppose a vector which represents caps a bottles:

$$\begin{bmatrix} b \\ c \end{bmatrix}$$

If we can spend two empty bottles and receive a full drink, that is equivalent to spending two bottles and getting one bottle and one cap. We will represent this operation as the addition of the following vector;

$$\begin{bmatrix} -2+1 \\ 1 \end{bmatrix}$$

And since we can drink before we pay, having only one empty bottle is enough to drink.

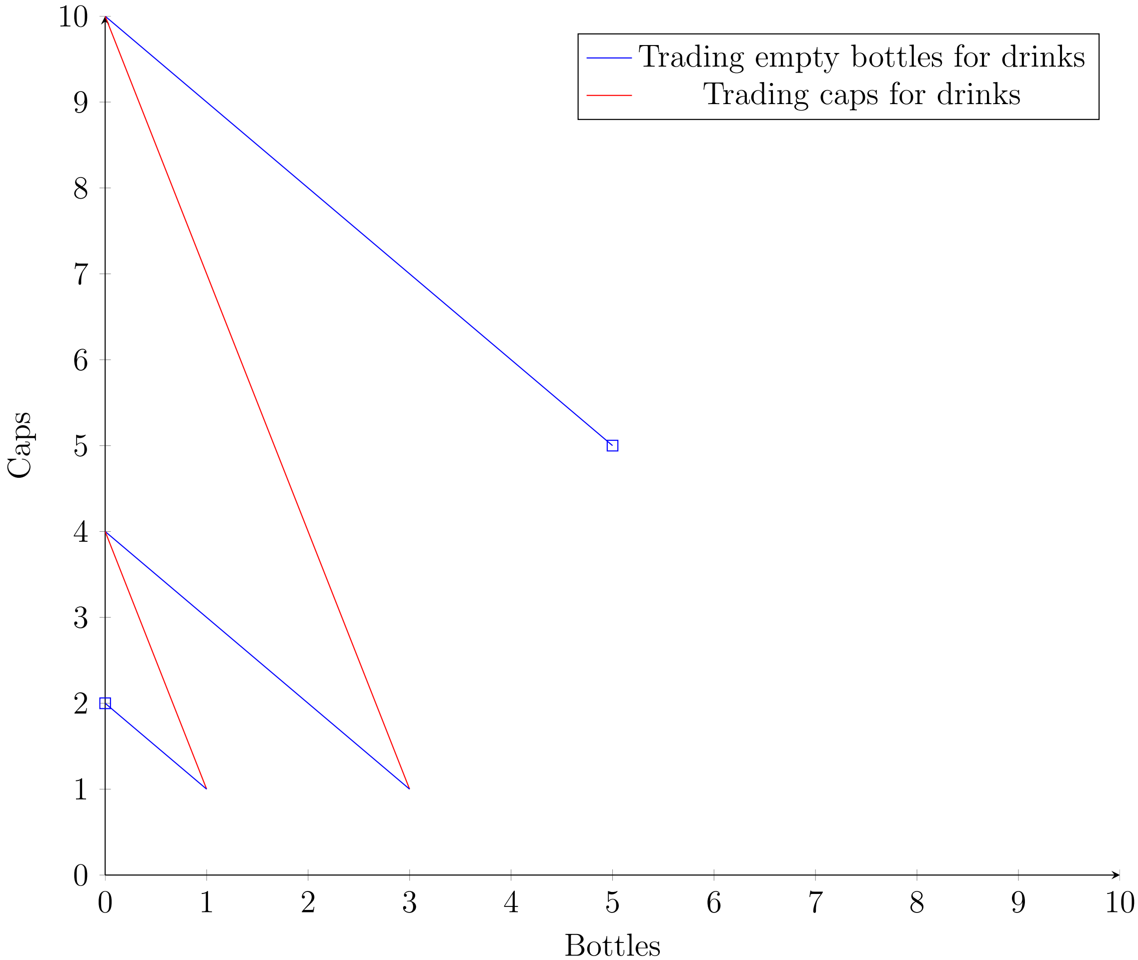
$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Using 4 caps can be represented similarly

$$\begin{bmatrix} 1 \\ -4+1 \end{bmatrix} = \begin{bmatrix} 1 \\ -3 \end{bmatrix}$$

4 Solution from special case

The two operations can be represented geometrically as a translation of a point on a 2 dimensional Cartesian plane. For example, if we start with 5 empty bottles and 5 empty caps, we can trace the motion of the point as following.



Since having either 1 bottle, or 3 caps is enough to do another transaction, we know from the context of the problem, that the final state must be one of the following points:

$$(0,0)$$

$$(0,1)$$

$$(0,2)$$

Visually, you can already tell that it's impossible to get to the point $(0,0)$ unless if you have already started there (sorry mate, you gotta buy some beer to play the game).

You can also tell that you cannot get to $(0,1)$, because that means you came from $(1,0)$ through a blue line, but that means you came from $(0,2)$ from a red line, which is impossible because $(0,2)$ is not enough to continue a transaction.

So already from this graph, you can tell that you will always end up with 2 caps left over (if we ignore the trivial case that you do not buy beer in the first place).

5 Algebraic Solution

Let $\vec{i}$ be the initial state, $\vec{f}$ be the final state, c be the number of cap-based transactions, and b be the number of bottle-based transactions.

$$\begin{aligned}
\vec{i} + c\begin{bmatrix}-3\\1\end{bmatrix} + b\begin{bmatrix}1\\-1\end{bmatrix} &= \vec{f}\\
c\begin{bmatrix}-3\\1\end{bmatrix} + b\begin{bmatrix}1\\-1\end{bmatrix} &= \vec{f} - \vec{i}\\
\begin{bmatrix}-3 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}c\\b\end{bmatrix} &= \vec{f} - \vec{i}\\
\begin{bmatrix}c\\b\end{bmatrix} &= \frac{1}{2}\begin{bmatrix}-1 & -1\\-1 & -3\end{bmatrix}(\vec{f} - \vec{i})
\end{aligned}$$

Initially we get the same amount of caps and bottles according to how many drinks i that we start with.

$$\begin{aligned}
\begin{bmatrix}c\\b\end{bmatrix} &= \frac{1}{2}\begin{bmatrix}-1 & -1\\-1 & -3\end{bmatrix}(\vec{f} - \vec{i})\\
\begin{bmatrix}c\\b\end{bmatrix} &= \frac{1}{2}\begin{bmatrix}1 & 1\\1 & 3\end{bmatrix}(\vec{i} - \vec{f})\\
\begin{bmatrix}c\\b\end{bmatrix} &= \frac{1}{2}\begin{bmatrix}1 & 1\\1 & 3\end{bmatrix} i\begin{bmatrix}1\\1\end{bmatrix} - \frac{1}{2}\begin{bmatrix}1 & 1\\1 & 3\end{bmatrix}\vec{f}\\
\begin{bmatrix}c\\b\end{bmatrix} &= i\begin{bmatrix}1\\2\end{bmatrix} - \frac{1}{2}\begin{bmatrix}1 & 1\\1 & 3\end{bmatrix}\vec{f}
\end{aligned}$$

If we simply make $\vec{f} = \vec{0}$, then we don't reach a contradiction even though we know from looking at the graph that it's not possible. This is because there's always a way to get to $(0,0)$ if we choose to fall into debt, but since that's against the rules, we have to make a check that there exists a way to reach $\vec{f}$.

We can describe the validity v of transaction c and b using the following recursive function:

$$\begin{aligned}
v(c,b) &= (v(c-1,b) + v(c,b-1)) \times \underbrace{(\begin{bmatrix}-3 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}c\\b\end{bmatrix} + \vec{i} > \vec{f})}_{\text{1 if this is true, 0 if this is false}}\\
v(0,0) &= 1
\end{aligned}$$

Notice above, I use + as OR and × as AND.

If we brute force using a script:

```
#!/usr/bin/python3
import numpy as np
import matplotlib.pyplot as plt
```

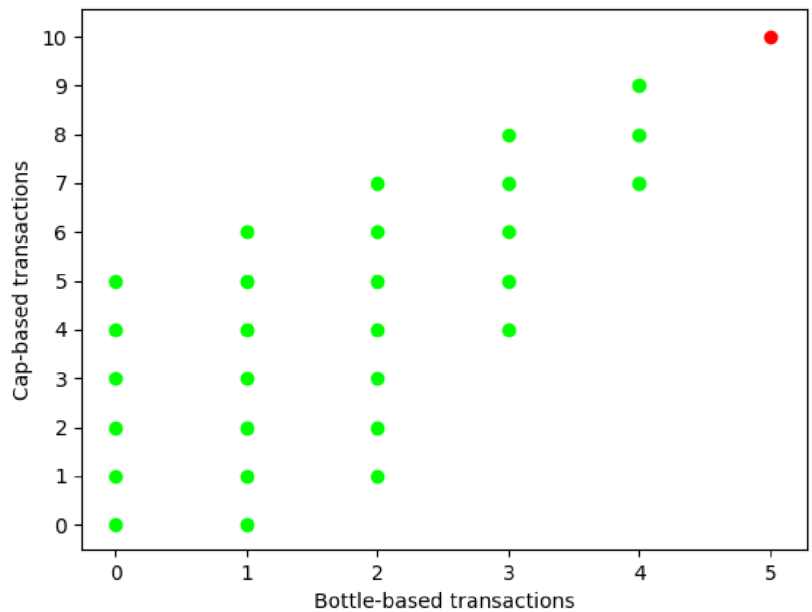
```python
t = np.array([[-3, 1],
              [1, -1]])

def transaction(a, b):
    return np.dot(t, np.array([[a],[b]])) + [[5], [5]]

ok = [] # ok points that have already been checked
check = [(0, 0)] # ok points, but may lead to more
tmp = []
while not len(check) == 0:
    for i in check:
        if not i in ok:
            ok.append(i)
        tran = transaction(i[0], i[1] + 1)
        if tran[0] >=0 and tran[1] >=0:
            tmp.append((i[0], i[1] + 1))
        tran = transaction(i[0] + 1, i[1])
        if tran[0] >=0 and tran[1] >=0:
            tmp.append((i[0] + 1, i[1]))
    check = tmp
    tmp = []

plt.scatter([i[0] for i in ok], [i[1] for i in ok], c=(0, 1, 0))
plt.scatter([5], [10], c=(1, 0, 0))
plt.xlabel('Bottle-based transactions')
plt.ylabel('Cap-based transactions')
plt.xticks(range(6))
plt.yticks(range(11))
plt.savefig('debt.png')
```

Then we get the following result where the green dots represents the possible transactions:



6 Using the final state of the vector to deduce the amount of drinks drunk

If we refer back to the naive solution, we concluded that beer itself is \$0.50. If we know what we have left over, we can deduce how much of our money was effectively turned into booze. If we started with \$10 dollars, and we are left with a bottles and b caps, then:

$$10 = 0.5x + a + 0.5b$$

$$x = \frac{10 - a - 0.5b}{0.5}$$

or

$$x = 20 - 2a - b \tag{1}$$